



#35  
12/12/03

THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the application of:

**Shimon GRUPER**

Examiner: Kenneth Tang

Serial No: **08/937883**

Group Art Unit: **2127**

Filed : **September 25, 1997**

For : **SOFTWARE APPLICATION ENVIRONMENT**

**RECEIVED**

DEC 08 2003

Technology Center 2100

**APPLICANT'S APPEAL BRIEF**  
**UNDER 35 U.S.C. §1.192**

Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Sir:

**(1) REAL PARTY IN INTEREST**

The real party in interest is Aladdin Knowledge Systems, the Assignee of the present application.

**(2) RELATED APPEALS AND INTERFERENCES**

There are no related appeals and interferences.

**(3) STATUS OF CLAIMS**

Claims 19 and 21-35 are pending and appealed.

**(4) STATUS OF AMENDMENTS**

No Amendments were filed subsequent to the Final Rejection.

**(5) SUMMARY OF INVENTION**

In the present application the term "learning period" refers to a period where there is a reasonable certainty that a software application is not infected by a virus. For example, assuming that a program was legally purchased, at the first period

following installation of the program on a user's machine there is a high certainty that the application is not infected by a virus. During this period the accesses of the application to data storage locations (files, folders, etc.) are registered (referred in the present application as an "enforcement file"), and they will be considered as the normal behavior of the program. After this "learning period" ends, any future access to files by the application will be tested against the registration enforcement file, and accesses that do not correspond to the normal behavior observed during registration will be considered and treated as suspect.

At page 9, lines 15-17, the present application refers to the "learning" issue "... the system allows the attempt and learns the details so that in future an access to that area of the disk will always be allowed. Thus a specific enforcement file is gradually built up over the duration of the learn mode."

For example, during two weeks from the installation time (the learning period) of an accounting program, it will be assumed that all of its accesses were to the C:\MyAccounting folder. According to the present invention, after the learning period is over no accesses to any other folder will be allowed.

Additionally, according to a further aspect of the present invention, in order to prevent program A, whose access is restricted by an "enforcement file", to invoke program B ("a daughter application" according to the present application terminology), which has no access limitations and therefore can reach any file/folder in the user's machine and cause damage, the restrictions of program A are "heritaged" to or imposed on program B. For instance, if a Web browser, which has an enforcement file, invokes a word processor, then the restrictions of the Web browser are applied to the word processor.

Thus, the present invention deals with the integrity issue by characterizing the normal access behavior of an application during a learning period, i.e., when the program is assumed to be uncorrupted, and restricting its accesses afterwards to what has been characterized as normal behavior.

**(6) ISSUES**

Are claims 19 and 21-35 unpatentable over US Patent 5,421,006 (Jablon)?

**(7) GROUPING OF CLAIMS**

Claims 19 and 21-35 stand or fall together.

**(8) ARGUMENTS**

Jablon addresses the integrity issue by indicating whether a program was altered since it was installed on a user's machine, and preventing altered programs from being executed. Indicating an unaltered program is carried out by "verification data" (e.g., a digital signature) of the program: "... the first program verifies the integrity of the second program using the verification data stored in the protectable non-volatile memory." (Col. 8, lines 48-50). In order to keep the verification data away from a malicious object (e.g., a virus), the digital signature is stored in a "protectable non-volatile memory". Thus, integrity means ensuring that a program has not been altered.

Jablon's invention uses a "hardware latch memory protection mechanism" (Col. 7, Lines 16-17). Its purpose is described as follows: "... the net effect of closing the latch is to prevent all subsequent programs from modifying the data .... BIOS will run the boot record program ..., and the system will continue its usual initialization process, with the restriction that the integrity assessment data cannot be modified. Thus, if the system subsequently changes the boot record, intentionally or not, the changes will be detected by BIOS the next time the system is reset." (Col. 12, lines 51-60).

The hardware latch memory enables storage of trusted data, whether it is a digital signature of the program or the program itself, and to restrict the access of the data to certain circumstances. "Before the first program transfers control to the second program, it computes, using known techniques, a modification detection code for the second program, and compares it to a pre-computed stored code in protectable non-volatile

memory. If the codes are not identical, then the second program is not run, and the user is warned of the program." (Col. 8, lines 57-64).

"One or more regions of non-volatile memory are provided, in which security-relevant data are stored. Access to a protectable memory region is controlled with a latch mechanism, such that the memory is always both readable and writable when the computer is first started. But during system initialization trusted software closes the latch to protect the memory, and thus prevent all subsequently run programs from reading and/or writing the security-relevant data during normal operation. Once closed, the latch can not be opened by software control. The latch is only re-opened when the system is restarted, which can occur by either momentarily turning off the power switch, or by pushing a reset switch. When the system is restarted, control of the CPU returns to a trusted startup program in read-only memory." (Col. 8, lines 18-33).

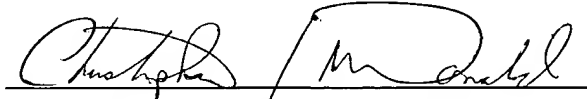
Jablon does not disclose a process for learning the normal access of an application for a limited time and, thereafter, using an enforcement device to prevent abnormal access by the application, as claimed in claims 19, 22, 24 and 25. Jablon doesn't examine the behavior of the program, but the content of the program's file(s). Thus, while Jablon examines if the program was changed, the present invention examines its behavior, that is, the normal access of the application.

Accordingly, Jablon does not disclose or suggest, either expressly or impliedly, the present invention is specifically set forth in Applicant's independent claims 19, 24 and 25. As such, those claims and the claims that may depend therefrom are believed to clearly and patentably distinguish the present invention from that taught by Jablon. Applicant thus kindly submits that the outstanding Section 102 rejection of claim 19 and the outstanding Section 103 rejection of claims 21-34, both in view of Jablon, are believed to be improper.

**CONCLUSION**

The claims are allowable over the prior art and it is respectfully requested that all rejections made by the Examiner be overturned and the application allowed to proceed toward issuance.

Respectfully submitted,

A handwritten signature in cursive script, appearing to read "Christopher J. McDonald", is written over a horizontal line.

Christopher J. McDonald  
Reg. 41,533

June 16, 2003

HOFFMAN, WASSON & GITLER, PC  
2361 Jefferson Davis Highway  
Suite 522  
Arlington, VA 22202  
(703) 415-0100

Attorney Docket No. A-7261.ab

Appendix of Claims

19. Apparatus for ensuring the integrity of an application executed on a computer having data storage arranged sectorwise, comprising:

apparatus for learning about the normal access behavior of said application to said data storage arranged sectorwise by monitoring accesses of said application to elements of said data storage during a limited period; and

an enforcement device, operative after said period is over, for identifying and preventing said application from accessing elements of data storage that do not correspond with the normal behavior of said application.

21. Apparatus according to claim 19 wherein said enforcement device is operative to prompt a user to give specific permission, upon occurrence of an attempt of the program to access files not accessed during said learning period.

22. Apparatus for ensuring the integrity of computer applications to be run in association with a computer having data storage arranged sectorwise in a storage device, comprising:

apparatus for assigning a general enforcement file to each new program;

apparatus for learning about the program by monitoring instances of user permission given to the program's attempts to make file accesses during a learning period; and

an enforcement device operative, after said learning period is over, to treat attempts of the program to access files to which the user permitted access during said learning period more leniently than attempts of the program to access files to which the user did not permit access during said learning period.

23. Apparatus according to claim 19 wherein said enforcement device is based at least partly on instances of specific permission being given by the user to the program to access certain files, wherein the enforcement device treats attempts

of the program to access files to which the user permitted access during said learning period more leniently than attempts of the program to access files to which the user did not permit access during said learning period.

24. Apparatus for ensuring the integrity of a computer application to be run in association with a computer having data storage arranged sectorwise in a storage device, comprising:

apparatus for assigning a general enforcement file to each new program;

apparatus for learning about the program by monitoring the program's attempts to make file accesses during a learning period;

an enforcement device operative, after said learning period is over, to treat attempts of the program to access files accessed during said learning period more leniently than attempts of the program to access files not accessed during said learning period, said enforcement device is based at least on instances of specific permission being given by the user to said application to access locations of said data storage, wherein said enforcement device treats attempts of said application to access locations of said data storage to which the user has permitted to access during said learning period more leniently than attempts of the program to access files to which the user did not permit access during said learning period.

25. A method for detecting abnormal behavior of a first application executed on a computer system, and preventing the damage thereupon, comprising:

- monitoring accesses of said application to elements of data storage arranged sectorwise in a storage device over a period of time and storing information about said accesses in an enforcement file, thereby learning the normal behavior of said application; and

- when said period is over, detecting attempts of said application to access elements of data storage that do not

correspond to said normal behavior as determined by said enforcement file and inhibiting said accesses, thereby preventing the damage thereupon.

26. A method according to claim 25, further comprising enabling the user of said first application to determine said normal behavior during said learning period.

27. A method according to claim 25, further comprising enabling the user of said first application to determine said normal behavior after said learning period is over.

28. A method according to claim 26, further comprising enabling the user of said first application to determine said normal behavior after said learning period is over.

29. A method according to claim 25, further comprising detecting attempts of a daughter application of said first application to access elements of data storage that do not correspond to said normal behavior as determined by said enforcement file and inhibiting said accesses, thereby preventing the damage thereupon.

30. A method according to claim 26, further comprising detecting attempts of a daughter application of said first application to access elements of data storage that do not correspond to said normal behavior as determined by said enforcement file and inhibiting said accesses, thereby preventing the damage thereupon.

31. A method according to claim 27, further comprising detecting attempts of a daughter application of said first application to access elements of data storage that do not correspond to said normal behavior as determined by said enforcement file and inhibiting said accesses, thereby preventing the damage thereupon.



32. A method according to claim 25, further comprising detecting attempts of a second application to access elements of data storage that do not correspond to said normal behavior as determined by said enforcement file and inhibiting said accesses, thereby preventing the damage thereupon.

33. A method according to claim 26, further comprising detecting attempts of a second application to access elements of data storage that do not correspond to said normal behavior as determined by said enforcement file and inhibiting said accesses, thereby preventing the damage thereupon.

34. A method according to claim 27, further comprising detecting attempts of a second application to access elements of data storage that do not correspond to said normal behavior as determined by said enforcement file and inhibiting said accesses, thereby preventing the damage thereupon.

35. A method according to claim 29, wherein said second application is executed on a second computer.